CLAIMS

I claim:

1. A method comprising organizing a byte stream of an information structure, said information structure having a schema and an in-memory representation, said schema having a schema tree representation with a plurality of schema nodes, said schema nodes including at least one leaf and at least one interior node, the step of organizing comprising the steps of:

computing a layout from the schema tree representation by depth-first enumeration of leaf nodes of the schema;

serializing the byte stream from the in-memory representation while grouping together all scalar items from the in-memory representation corresponding to each schema node; and

accessing information from the byte stream by using the layout and offset calculations.

2. A method as recited in claim 1, wherein said information structure is a message.

3. A method as recited in claim 1, wherein the step of computing a layout comprises:

establishing a fixed length portion of the byte stream, the fixed length portion having a slot for each enumerated schema leaf node; and

establishing a varying length portion of the byte stream following the fixed length portion, the varying length portion having successive areas for any information items requiring varying length encoding.

4. A method as recited in claim 1, wherein the step of computing a layout comprises:

establishing a fixed length portion of the byte stream, the fixed length portion having a slot for each enumerated schema leaf node having a predecessor in the depth-first numbering requiring varying length encoding; and

establishing a varying length portion of the byte stream following the fixed length portion, the varying length portion having successive areas for each enumerated schema node.

5. A method as recited in claim 1, wherein interior nodes of said schema tree representation are restricted to list and tuple nodes, and leaf nodes comprise scalar types and dynamic types.

6. A method as recited in claim 1, wherein the step of serializing the byte stream comprises:

determining a correspondence between the in-memory representation and the schema tree representation;

initializing the byte stream by reserving a fixed length portion and pointing to a beginning of a variable length portion;

retrieving a location in the byte stream for an element of the in-memory representation information corresponding to a first schema leaf node in depth first order from the layout;

converting the element to bytes in the byte stream according to a number of elements corresponding to the schema leaf node; and

repeating the steps of retrieving and converting for all schema leaf nodes in depth-first order.

7. A method as recited in claim 6, wherein the step of converting elements to bytes comprises recording a nested list of tuples in column order rather than row order, resulting in a set of nested lists.

8. A method as recited in claim 6, wherein the step of converting elements to bytes comprises preceding each list of varying length items with an offset table allowing any element of said each list to be reached in constant time from a head of said each list.

9. A method as recited in claim 1, wherein the step of accessing information comprises the steps of:

scanning a list of key values representing a table column serialized within the byte stream to determine an index position; and

using the index position in conjunction with offset calculations and offset tables serialized at the start of lists within the byte stream to find information in lists representing non-key table columns.

10. A method as recited in claim 1, wherein the schema tree representation is derived from a schema graph representation by truncating recursive definitions and variants and replacing truncated sub-trees with leaf nodes of a dynamic type.

11. A method as recited in claim 1, further comprising performing a preliminary reorganization of the schema to distribute tuples over variants prior to carrying out the steps of computing, serializing and accessing.

12. An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing organization of a byte stream of an information structure, the computer readable program code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 1.

13. A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for organizing a byte stream form of an information structure, said method steps comprising the steps of claim 1.

14. An apparatus comprising a serializer/deseralizer for a byte stream form of an information structure, said information structure having a schema and an in-memory representation, said schema having a schema tree representation with a plurality of schema nodes, said schema nodes including at least one leaf and at least one interior node, the serializer/deserializer comprising:

a processor for computing a layout from the schema tree representation by depth-first enumeration of leaf nodes of the schema;

a serializer for serializing the byte stream from the in-memory representation while grouping together all scalar items from the in-memory representation corresponding to each schema node; and

a selective de-serializer for accessing information from the byte stream by using the layout and offset calculations.

15. An apparatus as recited in claim 14, wherein the processor comprises a module for establishing a fixed length portion of the byte stream, the fixed length portion having a slot for each enumerated schema leaf node; and for establishing a varying length portion of the byte stream following the fixed length portion, the varying length portion having successive areas for any information items requiring varying length encoding.

16. An apparatus as recited in claim 14, wherein the processor comprises a module for establishing a fixed length portion of the byte stream, the fixed length portion having a slot for each enumerated schema leaf node having a predecessor in the depth-first numbering requiring varying length encoding; and for establishing a varying length portion of the byte stream following the fixed length portion, the varying length portion having successive areas for each enumerated schema node.

17. An apparatus as recited in claim 14, wherein the serializer comprises:

a reconciling module to determine a correspondence between the in-memory representation and the schema tree representation;

an initialization module to initialize the byte stream by reserving a fixed length portion and pointing to a beginning of a variable length portion;

a lookup module to retrieve a location in the byte stream for an element of the in-memory representation information corresponding to a first schema leaf node in depth first order from the layout;

a converter to convert the element to bytes in the byte stream according to a number of elements corresponding to the schema leaf node, wherein all schema leaf nodes are retrieved and converted in depth-first order.

18. An apparatus as recited in claim 17, wherein the converter comprises a recorder to record a nested list of tuples in column order rather than row order, resulting in a set of nested lists.

19. An apparatus as recited in claim 17, wherein the converter precedes each list of varying length items with an offset table allowing any element of said each list to be reached in constant time from a head of said each list.

20. An apparatus as recited in claim 14, wherein the selective de-serializer scans a list of key values representing a table column serialized within the byte stream to determine an index position, and uses the index position in conjunction with offset calculations and offset tables serialized at the starts of lists within the byte stream to find information in lists representing non-key table columns.

21. An apparatus as recited in claim 14, wherein the schema tree representation is derived from a schema graph representation by truncating recursive definitions and variants and replacing them with leaf nodes of dynamic type.

22. An apparatus as recited in claim 14, wherein a preliminary reorganization of the schema is performed to distribute tuples over variants prior to carrying out the remaining steps.

23. A computer program product comprising a computer usable medium having computer readable program code means embodied therein for causing organization of a byte stream form of an information structure, the computer readable program code means in said computer program product comprising computer readable program code means for causing a computer to effect the functions of claim 14.